Diagnosing Complex Software and Hardware

Brian C. Williams

First Year Report

April 2007

Introduction

Our first year got off to a slow start because of problems getting the JPL portion of the project funded. Since that problem was resolved we have made efforts to catch up. We have developed in prototype form the diagnosis infrastructure for diagnosing software and mixed hardware/software faults. This work is described below. Recently we met with the JPL team to explore the Earth Observing 1 (EO-1) space craft and its software infrastructure whose operation we wish to diagnose. Much work has recently been done to restructure the RMPL language, used for modeling spacecraft systems, and we will soon begin a dual thrust of:

1. Improving the technology readiness of the prototype code by incorporating the new RMPL front end and streamlining the integration of the stages of the diagnosis process; and
2. Developing models of the EO-1 software structure leading to a significant software diagnosis test/demonstration.

Mode Estimation and Diagnosis for PHCA

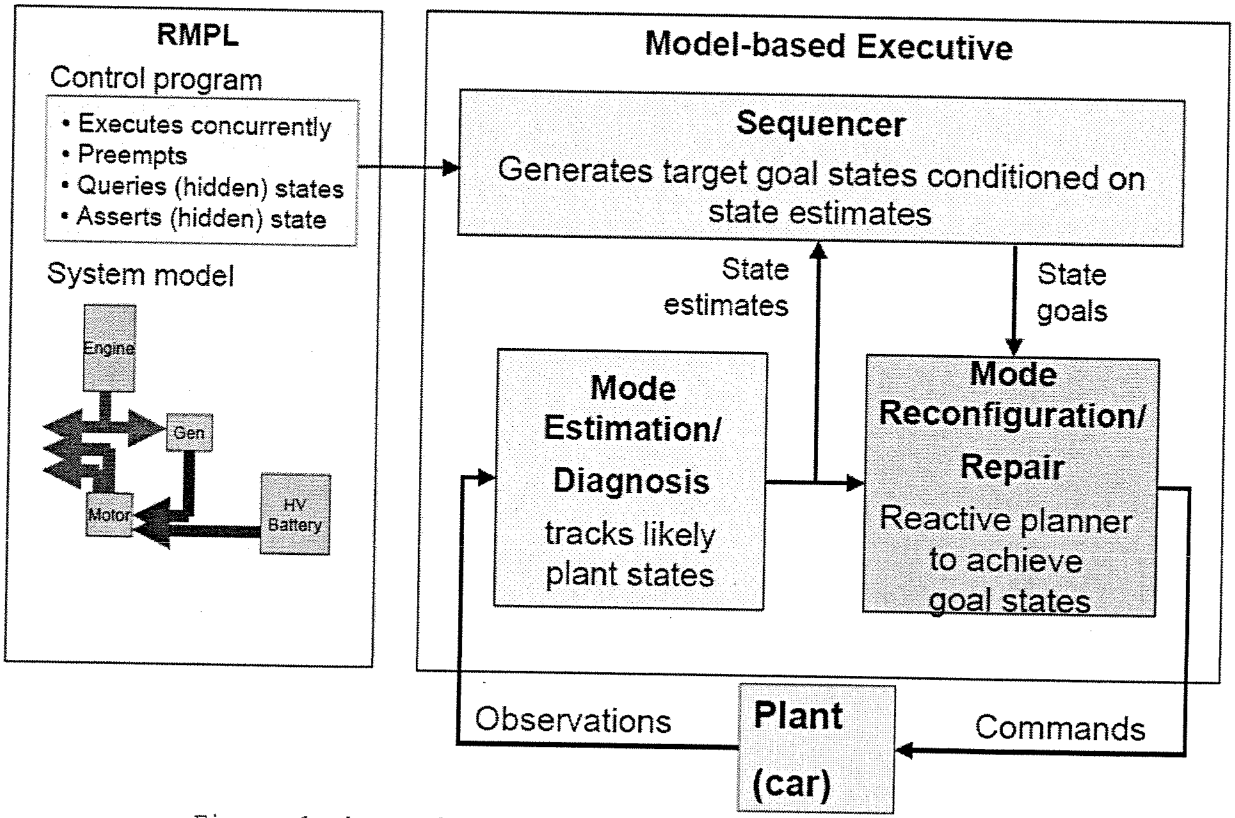
RMPL
Control program
• Executes concurrently
• Preempts
• Queries (hidden) states
• Asserts (hidden) state
System model
Engine
Gen
Motor
HV Battery
Model-based Executive
Sequencer
Generates target goal states conditioned on state estimates
State estimates
State goals
Mode Estimation/ Diagnosis
tracks likely plant states
Mode Reconfiguration/ Repair
Reactive planner to achieve goal states
Observations
Plant (car)
Commands

Figure 1 shows the model-based executive architecture.

The Mode Estimation and Diagnosis component estimates the most likely states of the system based on observations from the system and commands issued to the system. Given the current state of the system, the Mode Reconfiguration and Repair algorithm generates control actions in order to transition the system from its current state to a target state. The target state is referred to as a goal state, which can be input to the system by a higher level planner (sequencer, or driver). Both the diagnosis and reconfiguration algorithms operate based on models of components in the system (referred to as plant model).

Hierarchical Constraint Automata (HCA) were introduced as powerful representations of such behaviors. This capability has the advantage that HCA's are expressive models that can support modeling of complex systems consisting of both hardware and software components such as those found in the Earth Observer 1 (OE-1) platform.

We have built a prototype diagnosis engine that utilizes the proposed PHCA plan representation.

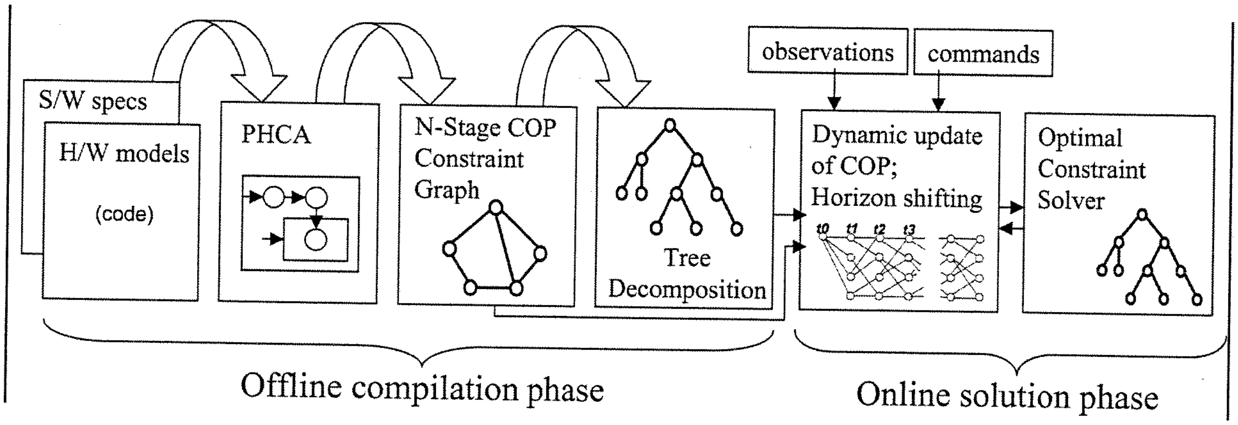
S/W specs
H/W models
(code)
PHCA
N-Stage COP Constraint Graph
Tree Decomposition
observations
commands
Dynamic update of COP; Horizon shifting
t0
t1
t2
t3
Optimal Constraint Solver
Offline compilation phase
Online solution phase

Figure 2: PHCA based diagnosis engine

Figure 2 shows the diagnostic system architecture proposed in our original proposal and implemented in our prototype.

The main features of this capability are as follows:

First, the diagnostic system is based on the expressive PHCA modeling formalism for capturing complex behaviors. PHCA-based diagnosis is defined as the task of enumerating and tracking the K most likely PHCA state trajectories.

Second, the diagnostic task is posed over a finite time horizon, in order to account for delayed symptoms.

Third, diagnosis is framed as a soft constraint optimization problem that encodes the PHCA models and their execution semantics over the finite horizon. This formulation integrates the PHCA simulation phase into the COP, thereby reducing the diagnosis task to that of solving the COP and tracking the most likely solutions, which correspond to the most likely PHCA state trajectories.

Fourth, tree decomposition is applied to the COP to decompose it into independent sub-problems by exploiting the structure of the constraints, thereby enabling the use of efficient optimal constraint solvers.

The PHCA-based diagnostic system architecture consists of two phases: an offline compilation phase and an online solution phase, as shown in Figure 2. The offline compilation phase precedes the online monitoring and diagnosis phase, and does not require the availability of any observations or issued commands. The following sections describe the modeling framework for diagnosing complex software and hardware systems, and the phases of the diagnostic architecture depicted in Figure 2.

The N-BFTE Process

The N-BFTE process, as illustrated in Figure 2, consists of two phases: an offline compilation phase and an online estimation phase. The following sections describe each phase.

Offline Phase

As illustrated in Figure 2, the offline compilation phase consists of three stages that specify the PHCA models, formulate diagnosis as a COP over a finite (N-Stage) time horizon, and apply tree decomposition to the COP. The offline phase sets the infrastructure for the online monitoring and diagnosis phase, and enables the efficiency of the online process. Observations and issued commands are not required for the offline phase; those are incorporated into the online diagnostic process described later.

In the offline phase, specifications of complex, software-extended behavior are compiled to PHCA models. These specifications are coded in the high-level Reactive Model-based Programming Language (RMPL).

Given PHCA models, diagnosis is defined as the task of enumerating and tracking the most likely PHCA state trajectories within a finite, N-Stage time horizon. The finite-horizon diagnosis accounts for delayed symptoms. The PHCA state trajectory estimation task is formulated as a soft constraint optimization problem (COP) within the N-Stage horizon.

The COP encodes the PHCA models and their execution semantics as probabilistic (soft) constraints, such that the optimal solutions correspond to the most likely PHCA state trajectories.

Encoding the PHCA execution semantics as constraints eliminates the need for a separate model simulation step during the online phase. Furthermore, soft constraints provide convenient expressivity for encoding the models, by not limiting uncertainty specification to decision variables.

Finally, tree decomposition is applied to the constraint network to exploit the independence of sub-problems through local consistency and dynamic programming, thus enabling the use of efficient optimal constraint solvers during the online phase.

Online Phase

The online phase uses the offline COP formulation and its tree decomposition, to enumerate and track the K most likely PHCA state trajectories that are consistent with observations and commands within the shifting N-Stage horizon. Recall that the COP generated offline does not specify assignments to observations and commands; those are available in the online phase. Therefore, the COP must be updated dynamically in the online phase.

The COP is updated in the online phase by shifting the N-Stage time horizon, incorporating new observations and commands, and inserting constraints for tracking the trajectories found within the previous horizon. Within each time horizon, the COP is solved using an efficient, decomposition-based optimal constraint solver. The solutions to the COP are enumerated in best-first order, thus efficiently focusing on the K most likely trajectories while allowing anytime behavior.

Decreasing the number of trajectories K being tracked solves the delayed-symptom problem by maintaining a larger number of states at each time step. However, for a system with many combinations of similar failure states with high probability, the number of trajectories maintained will have to be very large in order to be able to account for a delayed symptom that supports an initially low probability state. For such systems, considering even a small number of previous time steps N gives enough flexibility to regenerate the correct diagnosis.

RMPL Reactive Model Based Programming Language

A new implementation of the RMPL language was developed that supported the PHCA models utilized by the new diagnostic engine. This replaces the earlier RMPL implementation that was based on concurrent constraint automata (CCA).

The prototype diagnosis engine has been successfully demonstrated on some simple text cases. The diagnosis engine is presently being reworked for integration into the ASE engine so that we can begin testing the system with models of OE-1 software.

One meeting took place in February 2007 at JPL with Steve Chien to plan the integration efforts.

Conclusions

The new diagnosis engine and RMPL compiler are working well and match design expectations. We are somewhat behind schedule due primarily to a late start of JPL on the project and are we trying to catch up. We are expecting to have some encouraging results in year 2.